Towards Large-Scale Agent-Based Geospatial Simulation

Umar Manzoor[1] [0000-0001-7602-1914], Hamdi Kavak[2] [0000-0003-4307-2381], Joon-Seok Kim[2] [0000-0001-9963-6698], Andrew Crooks[3] [0000-0002-5034-6654], Dieter Pfoser[2], Andreas Züfle[2], and Carola Wenk[4] [000-0001-9275-5336]

[1] University of Hull, Hull HU6 7RX, United Kingdom
`u.manzoor@hull.ac.uk`
[2] George Mason University, Fairfax, VA 22030, USA
[3] University at Buffalo, Buffalo, NY 14260, USA
[4] Tulane University, New Orleans, LA 70118, USA

Abstract. Agent-based geospatial simulations have become very popular and widely used in examining the social and cultural characteristics of populations. Well-known toolkits such as NetLogo or MASON generally have scalability limitations, especially when the model and underlying spatial infrastructure become complex. This paper presents a framework for simulating large-scale agent-based geospatial systems by integrating the multi-agent systems toolkit JADE with the MASON agent-based modeling framework and its GIS extension, GeoMASON. The proposed Java-based framework can simulate large areas with hundreds of thousands of agents. It allows for the studying the evolution of a population and its environment over time. Such a framework provides the essential first steps for scalable model execution without sacrificing the model generality.

Keywords: Large-scale geospatial simulation · Agent-based Modeling · MASON · Jade · GIS.

1 Introduction

In agent-based modeling and simulation (ABMS), a system is modeled as a collection of autonomous and collaborative agents where each agent senses its situation, makes decisions to fulfill its goals, and acts on the environment [17, 19, 8]. ABMS are increasingly being used to study a wide range of complex phenomena, especially where large numbers of agents (entities) are needed. Applications range from studying non-spatial systems including modeling tumor growth and social networks [27] as well as a plethora of spatial applications such as building evacuation [21], the spread of a diseases (e.g., [9]), and the emergence of slums in urban environments [22]. In addition to these applications, ABMS have been employed for traffic prediction [6, 20] and pedestrian models and indoor navigation [3, 14]. Others have used such a methodology as data generators when "real world" data is missing (e.g., [11]). One could argue that these existing simulations provide scalability, but lack the abstraction to be used beyond their specific domains. At the same time, computational social science has developed powerful toolkits to design agents and environments for discrete and continuous geospatial simulations (geosimulations) also including transportation networks (e.g., [12, 23]). Often designed for quantitative studies and behavioral analysis, these geosimulations provide abstractions, but do not offer scalability to large numbers of agents.

This paper fills this gap by proposing a scalable and general ABMS framework for geospatial simulations involving networks. We propose solutions for the parallelization of the single-threaded GeoMASON tookit [24] by employing the Java Agent Development Environment (JADE) [5] for the communication between threads. Illustrated in Figure 1, our resulting framework allows to scale different spatial scenarios to millions of agents. For this purpose, we divide the space of our agents into partitions, each handled by a separate thread of execution. The main challenge addressed in this work is the synchronization between these threads. An agent may move between spatial partitions, and may communicate with other agents located in other partitions. As these partitions are running asynchronously, we must ensure that the clocks of different simulations are synchronized to avoid race conditions.

The proposed framework is evaluated on an urban model [11] , which simulates simple patterns of life within an urban setting. The model has spatial network for agent movement and social network for maintaining social links. We compared the performance of GeoMason [24] and the proposed framework on different settings, and concluded from experimentation that the proposed framework is outperformed by GeoMason when the agent population is small whereas with an increasing agent population, our proposed framework outperforms GeoMason as the complexity and time taken in simulation step increases substantially.

2 Related Work

Modeling and simulation has revolutionized many scientific and engineering fields in the past two decades. Some of the first efforts were based on state machines and used spatially explicit models for urban studies [4]. In recent years, advances in computational infrastructure have allowed to simulate large numbers of interacting actors,in agent-based modeling and simulation. Abar et al. [1] present a comparative survey of 85 modelling and simulation tools for ABMS (including the well-known MASON, NetLogo, and Repast toolkits). The authors highlight the salient features and shortcomings, and also specify the application domain or scope of each. In terms of scalability, only three out of 85 toolkits reviewed support extremely scalable simulations, namely MATSim [10], PDES-MAS [25], and Repast HPC [7]. MATSim is a domain specific open source framework and is confined to agent-based transport simulations. The PDES-MAS framework is based on the Parallel Discrete Event Simulation (PDES) paradigm, where the simulation model is divided into a network of concurrently executive processes, each maintain / process a spatial shard of the entire simulation scenario.

This, however, is not a realistic assumption for geospatial simulations as partitions are not isolated from the rest of the “world”. Following this assumption, for instance, models with social network that requires agent-to-agent communication and interaction within different zones will suffer. Given that agent-to-agent communication/interaction is one of the hallmarks of ABMS [9], this assumption will limit developing realistic large-scale geospatial agent-based models [2]. Repast HPC is the high performance

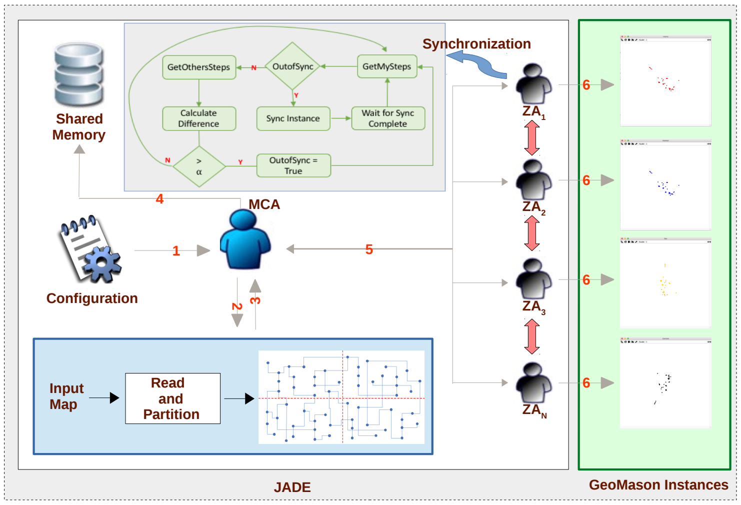


Fig. 1: System Architecture of Proposed Framework

cluster implementation of the Repast [7] toolkit which is an open source agent-based modeling and simulation platform. While it supports social network simulation and provides the ability to integrate geospatial data, it requires re-implementation of the model specific to Repast.

GeoMASON [24] is the geospatial extension of MASON [15], which is a general-purpose simulation framework and designed to support a wide range of multi-agent simulation tasks ranging from swarm robotics to social complexity environments [16]. GeoMASON supports various geospatial data formats including the support for reading/writing shapefiles. Using GeoMASON, one can create medium- to large-scale geospatial simulations. Here, the scale depends on the complexity of the agent logic and the geometry of the spatial environment. The main challenge of MASON is that it supports only single process execution. On the other hand, the Java Agent Development Environment (JADE) is a FIPA-compliant agent platform, which simplifies the implementation of distributed multi-agent systems. This Java-based framework supports peer-to-peer communication, is platform-independent, can be distributed across heterogeneous machines (including mobile devices), and supports the movement of agents across machines [5, 18]. JADE is the most popular industry-driven FIPA-compliant agent platform in academic and industrial communities [13]. While JADE provides a heavyweight agent architecture, we combine its strength in agent-to-agent communication with MASON's lightweight agent modeling support to address the aforementioned limitations of each.

3 System Architecture

The proposed framework is based on the layered agent architecture [26] to support scalability, is implemented in Java, and considers agent-based models with the spatial domain. Our proposed framework partitioned the spatial domain into so-called zones, and each zone is controlled by a separate instance of GeoMason. When agents leave the boundaries of their zones, they have to be passed from one GeoMason simulation to another. Analogously, when agents intend to communicate with agents located in another zone, their message has to be passed from one simulation to another.

To facilitate this communication, we implement each zone as a separate thread of execution. Threads are orchestrated by the JADE multi-agent system to handle communication. Thus, each JADE agent handles its own instance of GeoMason, which handles a potentially large number of simulation agents. We implement this functionality using two types of Jade Agents: *Zone Agents* and a *Master Controller Agent* (MCA). A model can have multiple Zone Agents, each running a GeoMason model instance for one particular zone. There is only one MCA that sets up the initial resource distribution and provides the communication among the zones, as shown in Figure 1. The remainder of this section provides more details for these two types of Jade Agents.

3.1 Master Controller Agent

MCA is the core agent that initializes and manages the whole system. During initialization, the MCA performs the following steps:

- It reads the configuration file which contains the application parameters such as the number of zones (N), number of agents per zone (M), etc.
- It creates and initializes the global shared memory.
- The MCA loads information about the spatial domain D and stores it in global shared memory. It then calls the World Partitioning Module which splits D into M zones.
- It creates and initializes the Global Statistics Module which is responsible for updating and displaying the global statistics of the application.
- The MCA creates and initializes N Zone Agents. It passes a unique zone identifier ($ZoneID$), the number of agents to be created in this Zone Agent (M), and a reference to the shared memory to each Zone Agent.

After initialization, MCA waits for updates from each Zone Agent and constantly monitors the status of these agents. In case of any error in any Zone Agent, it logs the error report, stops the simulation and re-runs the simulation from the last common checkpoint.

World Partitioning Module This module takes the spatial domain D and the number of desired zones (N) as input and partitions the spatial domain D into N zones. In our experiments we partition D into N parts of equal spatial size. In particular, we consider the example of a spatial domain with an underlying movement network that agents use for movement. While in our experiments we split the movement network using a simple spatial partition as shown in Figure 2, the World Partitioning Module could implement more sophisticated graph partitioning schemes that might aim to minimize the number of edges along the cuts.

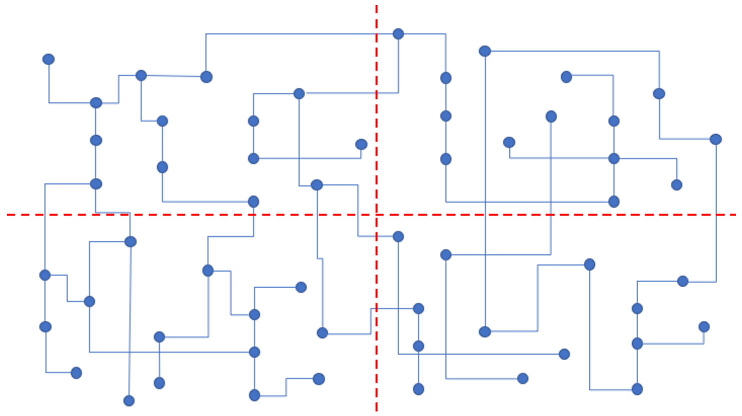


Fig. 2: Equal Split Spatial Network

Global Statistics Module This module is responsible for updating and displaying the statistics of the running simulation. It fetches all the required information from the global shared memory, and periodically updates the information. For example, the Global Statistics Module, can display the number of agents , buildings, and the current simulation step within the zone instance in the examples that follows (Section 4). However, the user can customize and display other information such as number of homes, offices, restaurants etc, if required.

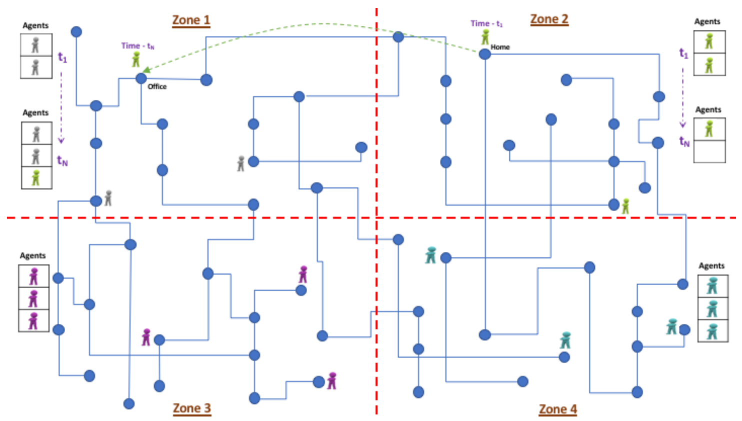


Fig. 3: Agent transfer between Zones

3.2 Zone Agent

Following initialization, each Zone Agent initializes a GeoMason instance for its zone. This instance can be initialized using a GUI or without (headless) depending on a parameter setting. The Zone Agent is responsible for keeping its instance synchronized with the other instances running in other Zone Agents. A cyclic behavior is added to each Zone Agent, which periodically calls the *Synchronization Module* to ensure that all Zone Agents are fully synchronized. Each Zone Agent keeps track of all the agents running in its GeoMason instance. These agents can move between zones, either by traveling between zones or by switching zones permanently. If an agent A moves from zone Z_2 to zone Z_1, then the Zone Agent for Z_2 performs the following steps: (1) It informs the Zone Agent for Z_1 about A's arrival time, (2) it changes A's status to *transit*, and (3) it calls the *Zone Transfer Module* to move A to Zone Agent Z_1, see Figure 3. After every C steps, where C is a configurable parameter, the Zone Agent creates a checkpoint for its local simulation, which is used by the MCA to re-run the simulation in case of an error.

Synchronization Module Each Zone Agent has Synchronization Module, which is responsible for synchronizing the given zone with others as shown in Figure 1. It performs the following steps: (1) Compute the difference D between the maximum simulation step of the other Zone Agents and the current simulation step of this Zone Agent. (2) Check if 'OutofSync' is true, go to step 4. (3) If D is greater than a pre-defined threshold, inform the Zone Agents by setting the 'OutofSync' flag in the shared memory to 'true'. (4) If D is greater than zero, run the simulation for D steps, and once complete, set 'SyncDone' to true for this ZoneAgent. (5) Wait until 'AllSync' is true, and then resume the simulation. ('AllSync' is set to true when all Zone Agents set 'SyncDone' to true.)

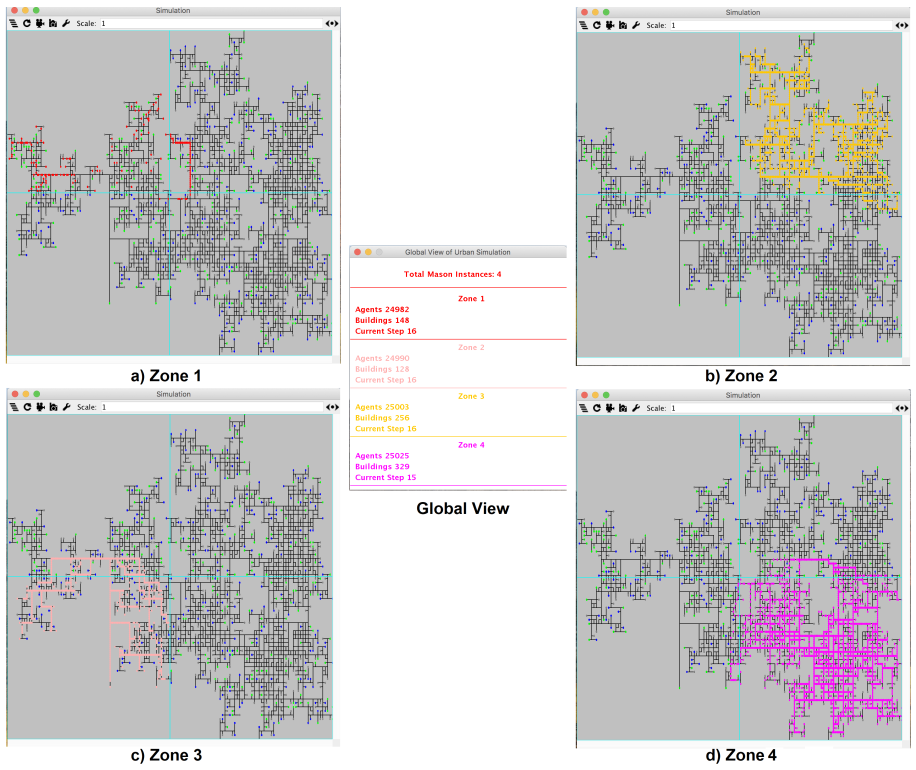
Zone Transfer Module This module takes three arguments: Agent A, the initial zone Z_i, and the target zone Z_t that A is moving to. This module performs the following steps: (1) Update A's location in the global shared memory. (2) Remove A from the instance scheduler and agent list in Z_i's GeoMason instance. (3) Send a message to Z_t by passing A's object as an argument. Z_t, in turn, adds the agent to its instance scheduler and agent list and sends an acknowledgment to Z_i when it is done.

4 Experimental Evaluation:

We evaluated our framework on the Urban Life model [11] which simulates simple patterns of life within an urban setting. The model has a spatial network for agent movement and assumes that each agent 1) has a home, 2) has a job and goes to work five days a week, 3) have friends, 4) has attributes (the most important one is happiness), 5) goes to the bar on weekends and a goal (maximize its happiness). Initially, each agent in the model is assigned a random home and work location, but during the simulation, the agent can decide to change home or work locations. During the simulation of a day, agents go to work in the morning, following the shortest path. After an (agent-specific) time at work, agents then return home on the shortest path.

The urban model has two types of social networks: a friends network and a work network. Initially, an agent is connected to every other agent sharing the same home (work) location in their friends (work) network. In this highly stylized model, each agent also has a dynamic attribute “happiness”. The happiness of an agent increases, when the number of social connections (of the corresponding friends and work networks) that are in the agent's proximity, is in a certain range. Thus, having too few or too many friends/co-workers around will make the agent sad. The minimum and the maximum number of this "sweet spot" is user-specified. Should the agent's happiness reach zero, the agent will "relocate" and will randomly select new work and home place in its vicinity while resetting his happiness to the initial value.

The experimental results reported in this section are collected using an Alienware laptop (Processor: 2.2 GHz Intel Core i7, RAM: 16GB, Graphics: Intel HD Graphics 1536 MB, OS: Microsoft Windows 10 Pro). The spatial network contains $51,309$ nodes, $131,076$ edges (average degree per node is 2.55), 65538 segments. Each GeoMason instance is aware of the spatial network to simplify shortest path computation. However, each zone knows its boundary. The social network is global, shared between each instance, and synchronized every night. Using GeoMASON with our JADE extension, the maximum number of agents that we can run is about $4x$ when compared to the standalone GeoMASON simulation. The framework provides separate visuals for each zone showing agents moving throughout the "day". Agents might pass from one zone to another. Figures 4 a-d show screenshots of zone-specific visualizations, whereas statistics for each zone are shown in an additional window. The framework also supports headless runs.



```
Total Mason Instances: 4

Zone 1
Agents 24982
Buildings 148
Current Step 16

Zone 2
Agents 24990
Buildings 128
Current Step 16

Zone 3
Agents 25003
Buildings 256
Current Step 16

Zone 4
Agents 25025
Buildings 329
Current Step 15
```

Fig. 4: Simulation using proposed architecture.

Table 1: GeoMason (GM) vs Jade-GeoMason (JM) with Hard Synchronization(HS) and Soft Synchronization(SS)

Agent Population	GM Time	JM Time(HS)	JM Time(SS)
$50,000$	9.82m	12.48m	8.71m
$100,000$	23.14m	25.09m	15.31m
$500,000$	132.8m	127.2m	111.2m
$1M$	-	218.41m	191.9m

4.1 Scenario 1

We compared and evaluated the performance of GeoMason and Jade-GeoMason (with four zones) on different agent population with hard synchronization (the GeoMason instances are synchronized whenever the step difference between the instance is more than one a) and soft synchronization (the GeoMason instances are not synchronized unless the step difference reaches a max difference threshold (configurable) or the agent is transferring from one zone to another). The initialization process is the same, i.e., agent work or home location can be in different zones, and either hard sync (i.e., the simulation step for each GeoMason instance should be the same) or soft sync (i.e., the GeoMason instances are only synchronized when the agent is transferring from one zone to another) is applied. The social network in Jade-GeoMason is global and synced once at night (simulation time). In this scenario, the agents frequently move between zones while traveling. The proposed framework with hard synchronization is outperformed by GeoMason when the agent population is small, i.e., one instance is sufficient since our model has a synchronization overhead. However, when an increasing agent population, our proposed framework outperforms GeoMason as the complexity and time taken in the simulation step increases substantially, as shown in Table 1. Furthermore, simulation with soft sync takes less time because of less synchronization overhead as compared to hard sync. Moreover, GeoMason is unable to simulate more than 600K agents for the Urban Life model, whereas Jade-GeoMason successfully simulated 1M agents.

4.2 Scenario 2

In this scenario, we evaluated the effects of different synchronization mechanisms and agent locations (home and work) on the performance of our proposed framework. Instead of hard synchronization, the instances are not synchronized unless the step difference reaches a maximum difference threshold (configurable), or the agent is transferring from one zone to another; we named this mechanism “soft synchronization” (SS). The results for this scenario are shown in Table 2. Restricting the agents’ home and work locations within the zone (ZM) significantly reduces hard synchronization time. Therefore, if more sophisticated graph partitioning schemes that might aim to minimize the number of edges along the cuts are used for partitioning (i.e. reduces transfer between zones), further performance boost for the framework could be achieved. Combining soft synchronization with zone-level location restriction significantly reduces the total simulation time as unnecessary synchronization is avoided and transfer between zones is minimized.

Table 2: Runtimes using Soft Synchronization (SS) and forcing work and home location within the same zone (ZM).

Agent Population	SS	ZM	SS + ZM
50, 000	8.71m	7.41m	5.12m
100, 000	15.31m	12.80m	9.98m
500, 000	111.2-m	121.4m	77.51m

5 Discussion and Future Work

Many factors such as partitioning, model complexity, agent population, hard synchronization, cross-zone movement frequency can influence the performance of our framework. In general, the framework can outperform the GeoMason simulation if the model complexity is high and the agent population is large. However, more experimentation using different models is needed to evaluate the effect of these factors on performance. We plan on further evaluating the proposed framework using models, partitioning techniques, and parameter settings. The experimentation reported in this paper is performed on a single node, and we plan to distribute zone simulations over a cluster (as JADE supports network deployment) and evaluate the performance of the framework. While what is presented is a rather stylized version of patterns of life and how work and home networks evolve. We are currently working on adding more realistic movement patterns and social network generation into the model and adding more realistic movement rules and millions of agents.

Acknowledgment

This project is sponsored by the Defense Advanced Research Projects Agency (DARPA) under cooperative agreement No.HR00111820005. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

1. Abar, S., Theodoropoulos, G.K., Lemarinier, P., OHare, G.M.: Agent based modelling and simulation tools: A review of the state-of-art software. Computer Science Review **24**, 13–33 (2017)
2. Anderson, T., Dragićević, S.: Neat approach for testing and validation of geospatial network agent-based model processes: case study of influenza spread. International Journal of Geographical Information Science **34**(9), 1792–1821 (2020)
3. Asahara, A., Maruyama, K., Sato, A., Seto, K.: Pedestrian-movement prediction based on mixed markov-chain model. In: Procc. 19th ACM SIGSPATIAL. pp. 25–33. ACM (2011)
4. Benenson, I., Torrens, P.M., Torrens, P.: Geosimulation: Automata-based modeling of urban phenomena. John Wiley & Sons (2004)
5. Bergenti, F., Caire, G., Gotta, D.: Agents on the move: JADE for Android devices. In: Proceedings of the 16th Workshop ”From Objects to Agents”. CEUR-WS.org (2014)
6. Brinkhoff, T.: A framework for generating network-based moving objects. GeoInformatica **6**(2), 153–180 (2002)

7. Collier, N., North, M.: Parallel agent-based simulation with repast for high performance computing. Simulation **89**(10), 1215–1235 (2013)
8. Crooks, A., Malleson, N., Manley, E., Heppenstall, A.: Agent-based Modelling and Geographical Information Systems: A Practical Primer. Sage (2018)
9. Crooks, A.T., Hailegiorgis, A.B.: An agent-based modeling approach applied to the spread of cholera. Environmental Modelling & Software **62**, 164–177 (2014)
10. Horni, A., Nagel, K., Axhausen, K.W.: The multi-agent transport simulation MATSim. Ubiquity Press (2016)
11. Kim, J.S., Jin, H., Kavak, H., Rouly, O.C., Crooks, A., Pfoser, D., Wenk, C., Züfle, A.: Location-based social network data generation based on patterns of life. In: 21st IEEE International Conf. on Mobile Data Management. pp. 158–167. IEEE (2020)
12. Kim, J.S., Pfoser, D., Züfle, A.: Vehicle relocation for ride-hailing. In: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA). pp. 589–598. IEEE (2020)
13. Kravari, K., Bassiliades, N.: A survey of agent platforms. JASSS **18**(1), 11 (2015), http://jasss.soc.surrey.ac.uk/18/1/11.html
14. Kwak, S., Nam, H., Jun, C.: An enhanced indoor pedestrian model supporting spatial dbmss. In: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness. pp. 25–32. ACM (2010)
15. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: A multiagent simulation environment. Simulation **81**(7), 517–527 (2005)
16. Luke, S., Simon, R., Crooks, A., Wang, H., Wei, E., Freelan, D., Spagnuolo, C., Scarano, V., Cordasco, G., Cioffi-Revilla, C.: The mason simulation toolkit: past, present, and future. In: International Workshop on Multi-Agent Systems and Agent-Based Simulation. pp. 75–86. Springer (2018)
17. Macal, C., North, M.: Introductory tutorial: Agent-based modeling and simulation. In: Proce. Winter Simulation Conference 2014. pp. 6–20 (2014)
18. Manzoor, U., Nefti, S., Rezgui, Y.: Categorization of malicious behaviors using ontology-based cognitive agents. Data & Knowledge Engineering **85**, 40–56 (2013)
19. Manzoor, U., Zafar, B.: Multi-agent modeling toolkit–mamt. Simulation Modelling Practice and Theory **49**, 215–227 (2014)
20. Mokbel, M.F., Alarabi, L., Bao, J., Eldawy, A., Magdy, A., Sarwat, M., Waytas, E., Yackel, S.: Mntg: an extensible web-based traffic generator. In: International Symposium on Spatial and Temporal Databases. pp. 38–55. Springer (2013)
21. Niu, L., Song, Y.: A simulation model fusing space and agent for indoor dynamic fire evacuation analysis. Simulation **92**(3), 215–232 (2016)
22. Patel, A., Crooks, A., Koizumi, N.: Slumulation: an agent-based modeling approach to slum formations. JASSS **15**(4), 2 (2012), http://jasss.soc.surrey.ac.uk/15/4/2.html
23. Pesavento, J., Chen, A., Yu, R., Kim, J.S., Kavak, H., Anderson, T., Züfle, A.: Data-driven mobility models for covid-19 simulation. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities. pp. 29–38 (2020)
24. Sullivan, K., Coletti, M., Luke, S.: Geomason: Geospatial support for mason. Tech. rep., Department of Computer Science, George Mason University (2010)
25. Suryanarayanan, V., Theodoropoulos, G., Lees, M.: Pdes-mas: Distributed simulation of multi-agent systems. Procedia Computer Science **18**, 671–681 (2013)
26. Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
27. Zia, K., Farrahi, K., R, A., Ferscha, A.: An agent-based parallel geo-simulation of urban mobility during city-scale evacuation. Simulation **89**(10), 1184–1214 (2013)